LAPIN AMK
Lapland University of Applied Sciences

SILVERSTRIPE-SISÄLLÖNHALLINTAJÄRJESTELMÄ 3.2

Allan Haapalainen

Opinnäytetyö
Tekniikka ja liikenne
Tietotekniikan koulutusohjelma
Insinööri (AMK)

2015

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tekniikka ja liikenne
Tietotekniikan koulutusohjelma

Tekijä Allan Haapalainen Vuosi 2015

Ohjaaja Tauno Tepsa
Työn nimi Silverstripe-sisällönhallintajärjestelmä 3.2
Sivu- ja liitemäärä 31 + 0

Tämän opinnäytetyön tavoitteena oli luoda opas sekä tutoriaali Silverstripe-sisällönhallintajärjestelmästä kiinnostuneille sen asennuksesta, käyttöönotosta sekä perusohjelmoinnista. Lähteenä on käytetty pelkästään Silverstripe:n kotisivulla löytyvää kehittäjäopasta, koska kirjoitushetkellä versio 3.2 oli vain pari viikkoa vanha, ja uusimmat tiedot löytyivät ainoastaan Silverstripe:n omilla sivuilla.

Tämä opas käy läpi Silverstripe-projektin perusohjelmointiin liittyvät keskeisimmät osat sekä selittää DataObject-osan luomis- sekä käyttöönottoprosessin.

Aihe oli valittu kirjoittajan omasta mielenkiinnosta aiheeseen käytettyä sitä oman yrityksen projekteissa. Sisällönhallintajärjestelmistä tunnetuimmat ovat tällä hetkellä Wordpress ja Joomla, mutta Silverstripe on jäänyt käyttämättä monilta kehittäjiltä, koska se ei ole niin laajasti tunnettu vielä. Tämän työn tarkoituksena on antaa kehittäjille lisää vaihtoehtoja sisällönhallintajärjestelmän valitsemiseen seuraavaa verkkosivuprojektia varten.

Avainsanat Silverstripe, DataObject, CMS, Sisällönhallinta, PHP

Abstract of Thesis

LAPIN AMK
Lapland University of Applied Sciences

School of Technology, Communication and Transport
Information Technology Programme

Author Allan Haapalainen Year 2015

Supervisor(s) Tauno Tepsa
Subject of thesis Silverstripe Content Management System 3.2
Number of pages 31 + 0

The goal for this thesis was to create an introduction and a tutorial about Silverstripe content management system's installation, implementation and basic programming for the people interested in it. Silverstripe's own developer guide was used as the only source for this thesis because version 3.2 of the software was only a couple weeks old during the making of the thesis and the newest information about the software was solely found on Silverstripe's main website.

This guide goes through the most essential parts of Silverstripe regarding programming and also explains the creation and implementation process of making one of the essential parts called DataObject.

The subject was chosen based on writer's own interest in the subject after using it in his own business projects. The most known content management systems today are Wordpress and Joomla but Silverstripe has not been used by many developers just because it is not so widely known yet. The main purpose of this thesis is to give developers more choices in choosing the content management system for their next project.

Key words Silverstripe, DataObject, CMS, Content Management, PHP

SISÄLLYS

KUVIOLUETTELO

ESIMERKKIKOODI-LUETTELO

KÄSITELUETTELO

Admin	Järjestelmävalvoja
Apache	Avoimeen lähdekoodiin perustuva palvelinohjelmisto
CMS	Content Management System eli sisällönhallintajärjestelmä
Controller	Silverstripe-luokka, joka huolehtii http-pyynnöistä ja -vastauksista
CSS	Cascading Style Sheets, www-sivuston dokumentti, jossa määritellään sivuston elementtien tyylit
CSV	Comma-separated values on tiedostomuoto, joka sisältää taulukkomuotoista tietoa
DataObject	Silverstripe-luokka, joka määrittelee tietokannan sarakkeet, relaatiot ja muut tiedot
Front-end	Rajapinta, joka näkyy sivuston käyttäjälle
HTTP	Hypertext Transfer Protocol on protokolla, jota selaimet ja internet-palvelimet käyttävät tiedonsiirrossa
JavaScript	Dynaaminen komentosarjakieli, jota käytetään www-sivuissa
JSON	JavaScript Object Notation on tiedostosiirtoformaatti tiedonvälitykseen
MySQL	Relaatiotietokantaohjelmisto
Open source	Avoin lähdekoodi
ORM	Object-Relational Model on tapa, jota Silverstripe käyttää esittääkseen tietonsa
PHP	Hypertext Preprocessor on ohjelmointikieli, jota käytetään internet-palvelinympäristöissä
PhpMyAdmin	MySQL-tietokannan hallintatyökalu
SS	Silverstripe
XML	Extensible Markup Language, merkintäkieli
YAML	YAML Ain't Markup Language, merkintäkieli
Zip	Pakkausformaatti

1 JOHDANTO

Opinnäytetyön aiheena oli esitellä Silverstripe-sisällönhallintajärjestelmä ja sen keskeisimmät osat. Työn aihe oli valittu omasta mielenkiinnosta kyseiseen teknologiaan sen jälkeen, kun siihen perehdyin uudella työpaikalla.

Työn alkuosassa käydään läpi Silverstripe-alustan asennus ja perusprojektin rakenne, jossa esitellään verkkosivujen kehittäjälle tärkeimmät kansiot ja tiedostot. Seuraavaksi esitellään Silverstripe-projektille kenties tärkein ominaisuus eli DataObject, jolla pystyy laajentamaan perusprojektia ja tekemään siitä entistä dynaamisemman. Lopuksi käsitellään projektin Admin-paneelia, sen rakennetta ja miten saadaan Admin-paneelin tiedot näkymään front-endissä. Työn loppuosassa näytetään, miten tehdään oma DataObject ja miten se hyödynnetään sekä Admin-paneelissa että front-endissä.

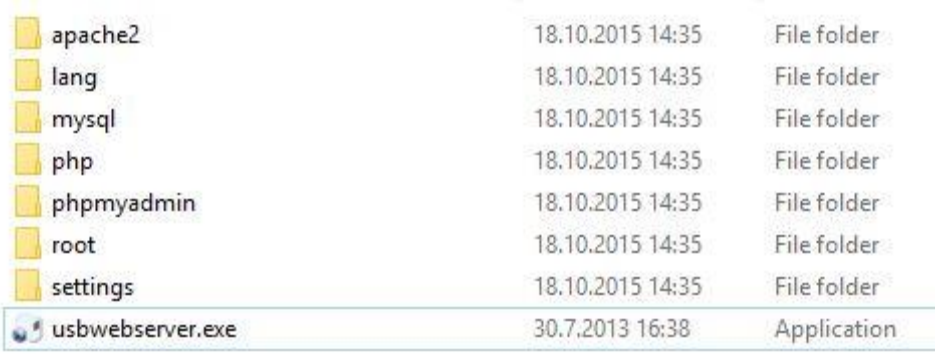
Opinnäytetyön tavoitteena oli luoda ohje, joka opastaa Silverstripe-teknologiasta kiinnostuneita sen käyttöönotossa ja esittelee keskeisimmät osat, jotta uudet käyttäjät voisivat alkaa tehdä omia Silverstripe-projekteja ilman tarkkaa perehtymistä.

2 SILVERSTRIPE-SISÄLLÖNHALLINTAJÄRJESTELMÄ

2.1 Asennus

Silverstripe-alusta voidaan ladata ilmaiseksi Silverstipen kotisivulta, koska se on open-source eli avoimen lähdekoodin ohjelma. Koodi on pakattu zip-tiedostoon, ja sen koko on vain 9.6 megatavua. Zip-tiedosto puretaan palvelimeen juurihakemistoon. (Silverstripe.org 2015a.)

Tässä opinnäytetyössä palvelimena on käytetty USBWebserverin versiota 8.6, jonka voi ladata myös ilmaiseksi osoitteesta http://www.usbwebserver.net/en/download.php. Zip-tiedosto sisältää kaikki palvelimen toimintaan tarvittavat komponentit kuten Apache, MySQL ja PhpMyAdmin (Kuvio 1). USBWebserver-tiedostot puretaan USB-tikulle.



Kuvio 1. USBWebserverin kansiorakenne

Kun USBWebserverin juurihakemiston eli root-kansion oletustiedostot on poistettu, voidaan siihen purkaa Silverstripe-kansion tiedostot. Sen jälkeen käynnistetään palvelin ajamalla usbwebserver.exe-tiedosto, jonka jälkeen käyttöjärjestelmä pyytää antamaan ohjelmalle oikeudet ajaa mysqld_usbwv8.exe-tiedosto sekä Apache HTTP Server (Kuvio 2). Kun oikeudet on annettu, valitaan palvelimen oletuskieli, jonka jälkeen voidaan asettaa tarvittavat asetukset Silverstripen asennusta varten.

Kuvio 2. Oikeuksien antaminen USBWebserverille

Seuraavaksi avataan settings-kansion php.ini-tiedosto ja vaihdetaan max_execution_time-muuttuja 30:stä 360:neen (Esimerkkikoodi 1). Sen jälkeen poistetaan kommenteista muuttuja date.timezone ja asetetaan arvoksi Europe/Helsinki (Esimerkkikoodi 2). Seuraavaksi poistetaan kommenteista myös laajennus fileinfo.dll (Esimerkkikoodi 3). Lopuksi kytketään pois päältä display_errors-muuttuja (Esimerkkikoodi 4). Tämän jälkeen käynnistetään palvelin uudelleen, jotta muutokset päivittyisivät palvelimelle. Sen jälkeen ajetaan selaimessa palvelimen osoite *localhost:8080*.

```
; Maximum execution time of each script, in seconds
; http://php.net/max-execution-time
; Note: This directive is hardcoded to 0 for the CLI SAPI
max_execution_time = 360
```

Esimerkkikoodi 1. max_execution_time-muuttujan muokkaus php.ini-tiedostossa

```
[Date]
; Defines the default timezone used by the date functions
; http://php.net/date.timezone
date.timezone = Europe/Helsinki
```

Esimerkkikoodi 2. date.timezone-muuttujan muokkaus php.ini-tiedostossa

```
;extension=php_bz2.dll
extension=php_curl.dll
extension=php_fileinfo.dll
extension=php_gd2.dll
```

Esimerkkikoodi 3. fileinfo.dll-laajennuksen lisääminen php.ini-tiedostossa

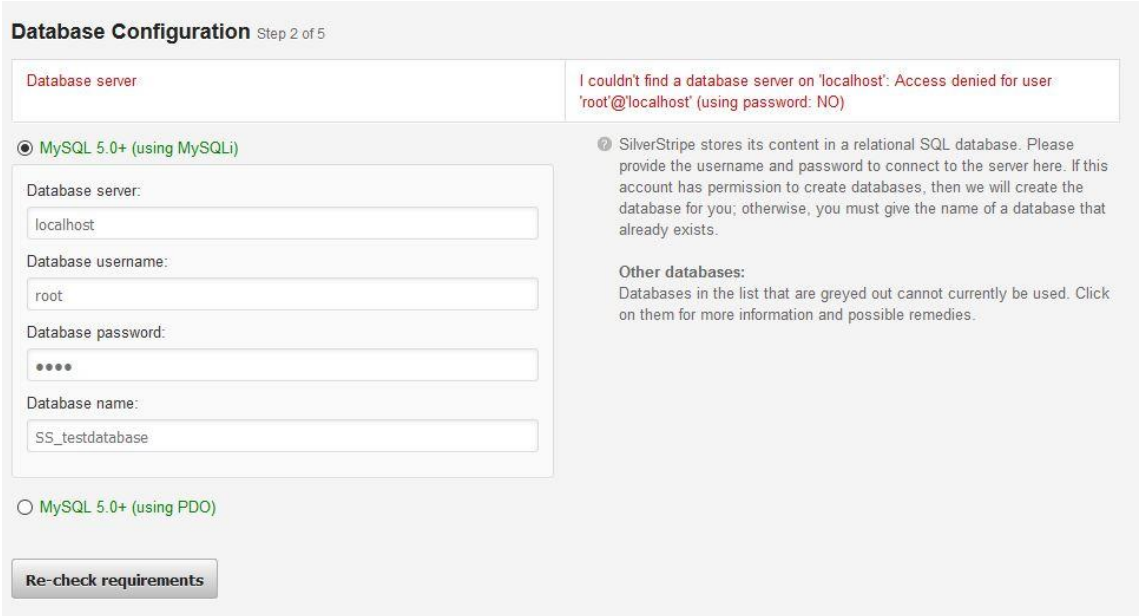
```
; Default Value: On
; Development Value: On
; Production Value: Off
; http://php.net/display-errors
display_errors = Off
```

Esimerkkikoodi 4. display_errors-muuttujan muokkaus php.ini-tiedostossa

Kun sivu latautuu, Silverstripe-asennustyökalu antaa virheilmoituksen, koska tietokantatietoja ei ole vielä määritelty (Kuvio 3). USBWebserverin oletustunnukset ovat:

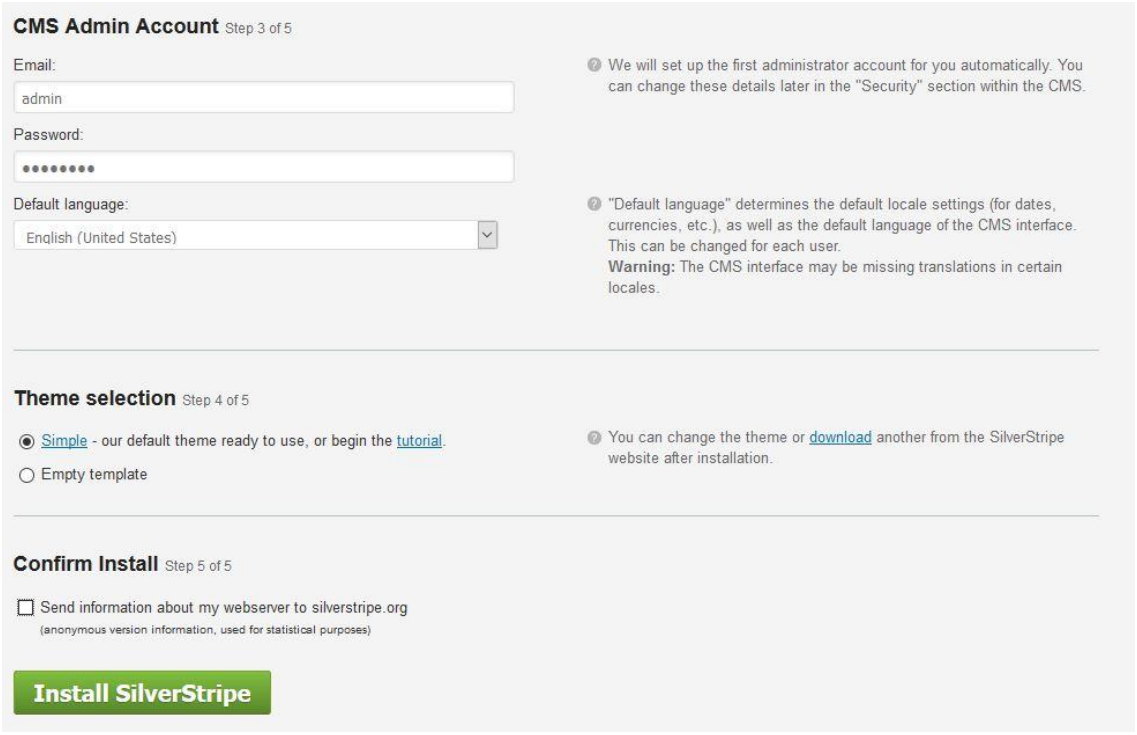
- palvelin: localhost
- käyttäjä: root
- salasana: usbw

Tietotokannan nimen voi keksiä itse, ja sen jälkeen painetaan Re-check requirements-painike, jolloin tiedot päivittyvät palvelimelle.



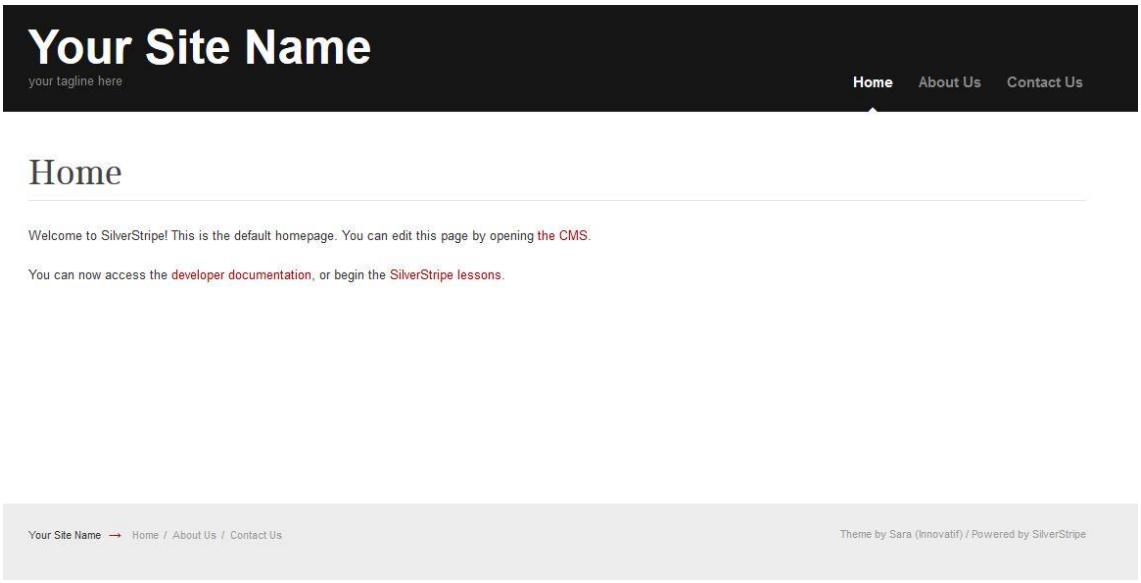
Kuvio 3. Tietokantatietojen syöttäminen

Sen jälkeen pitää syöttää Admin-paneelia varten käyttäjätunnus tai sähköposti, salasana ja oletuskieli. Seuraavaksi valitaan oletusteema ja painetaan Install Silverstripe-painiketta (Kuvio 4), jolloin asennusprosessi alkaa (Kuvio 5). Kun asennus on loppunut, poistetaan install.php-tiedosto root-kansiosta ja Silverstripe on nyt käyttövalmis. Ajamalla selaimessa osoite *localhost.8080* saadaan oletussivu näkyviin (Kuvio 6).



Kuvio 4. Admin-paneelitietojen syöttäminen

Kuvio 5. Silverstripen asennusprosessi



Kuvio 6. Silverstripen oletusteema

2.2 Silverstripe-projektin rakenne

Silverstripe-projektin rakenne voidaan jakaa neljään pääryhmään: ydinkansiot, kustomoitavat kansiot, teema-kansiot ja moduuli-kansiot (Kuvio 7). Tämä ra-

kenne perustuu periaatteeseen ”convention over configuration” eli koodauskäytännöt ovat etusijalla. Tällöin tiedostojen ja kansioiden paikoilla on yhteys niiden logiikkaan. (Silverstripe.org 2015b.)

Kuvio 7. Sliverstripe-projektin kansiorakenne

Ydinkansioihin kuuluvat assets-, cms-, siteconfig- ja framework-kansiot. Assets-kansioon tulevat kaikki Admin-paneelin kautta ladatut tiedostot, kuten kuvat, pdf-tiedostot jne. Assets-kansioon voi myös itse siirtää tiedostoja ja käyttää niitä myöhemmin Admin-paneelissa. Cms-kansioon kuuluvat kaikki Admin-paneelia koskevat tiedostot, ja Cms-kansioon rakenne muistuttaa mysite-kansiota. Siteconfig-kansio sisältää työkalun, jolla pystyy tekemään Admin-paneelissa koko sivustoa koskevia muutoksia. Framework-kansiossa ovat kaikki Silverstripe-alustalle ominaiset tiedostot, joita hyödynnetään sekä Admin-paneelissa että omissa sivuissa. (Silverstripe.org 2015b.)

Kustomoitavia kansioita on periaatteessa vain yksi eli mysite-kansio. Tähän kansioon tulevat kaikki omat PHP-, YAML- ja XML-tiedostot, kuten sivutyypit, Data-Objectit, kielitiedostot jne. Tätä kansiota kehittäjä tulee hyödyntämään todella usein tehdessään Silverstripe-projektia. (Silverstripe.org 2015b.)

Themes- eli teema-kansioon tulevat kaikki Silverstripen kotisivulta ladattavat teemat. Tämä on kenties toiseksi tärkein kansio, koska tähän tulevat kaikki front-endiä koskevat CSS-, JavaScript-, SS- ja fonttitiedostot sekä esim. staattiset kuvatiedostot. Templates-alikansio sisältää front-endille kaksi tärkeintä kansiota, jotka ovat Includes ja Layout.

Includes-kansioon tulevat kaikki SS-tiedostot, joita halutaan sisällyttää sivuaihioon. Tämä voi olla esimerkiksi sivuston ylä- ja alatunniste. Nämä tiedostot sisällytään aihioon käyttämällä syntaksia *<% include [includes-tiedoston nimi] %>.* Esimerkkikoodi 5 näyttää, miten tämä on toteutettu oletusprojektin Page.ss-tiedostossa. (Silverstripe.org 2015c.)

```
<% include Header %>
<div class="main" role="main">
    <div class="inner typography line">
        $Layout
    </div>
</div>
<% include Footer %>
```

Esimerkkikoodi 5. Includes-tiedoston sisällytys sivuaihioon

Layout-kansioon tulevat kaikki ss-päätteiset sivuaihiotiedostot. Näitä tiedostoja voi käyttää vain kerran ja Layout-kansiossa olevan tiedoston on oltava samanniminen kuin mysite-kansiossa olevan sivutyypin, jotta tiedot tulostuisivat oikealle sivulle. (Silverstripe.org 2015d.)

Moduuli-kansiot ovat kaikki kolmansien osapuolien tekemät lisäosat, joita voi hakea osoitteesta *http://addons.silverstripe.org/add-ons*. Niillä täytyy olla _config.php-tiedosto sekä _config-kansio sisällään, jotta ne olisivat oikeita moduuleita. Näiden kansioiden rakenne muistuttaa mysite-kansion rakennetta. Moduulit lisätään projektin juurihakemistoon, kuten esimerkiksi oletusprojektin reports-moduulikansio. (Silverstripe.org 2015c.)

2.3 DataObject ja ORM

Silverstripe käyttää niin sanottua ORM-tekniikkaa (object-relational model), jolla se esittää Silverstripen informaatiota. Tämä tarkoittaa, että oliot muutetaan relaatioiksi ja relaatiot muutetaan takaisin olioiksi. Tällöin jokaisella tietokannan taulukolla on oma PHP-luokka, jokaisella tietokannan rivillä on oma PHP-olio ja jokaisella tietokannan sarakkeella on muuttuja PHP-oliossa. (Silverstripe.org 2015e.)

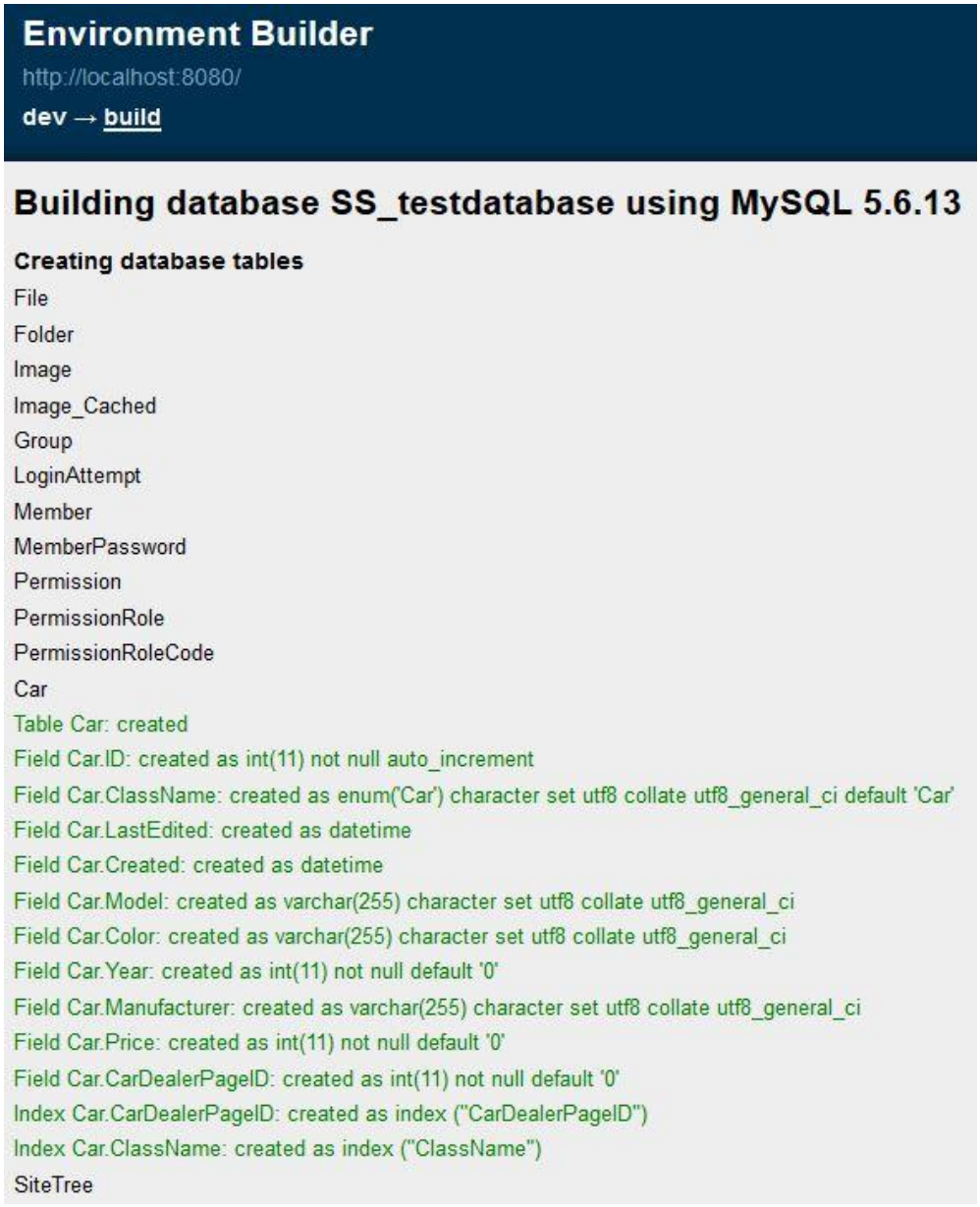
DataObjectin aliluokkia ovat kaikki Silverstripen tietotaulukot. Jokainen tietokannan rivi on luokkatyyppiä DataObject. Tietokannan sarakkeet ovat DataObjectin tietotyyppejä, jotka määritellään DataObjectin staattisessa $db-muuttujassa (Esimerkkikoodi 6). Jokaisella DataObjectilla täytyy olla myös määriteltynä yksi tai useampi relaatio, esim. johonkin tiettyyn sivutyyppiin. Relaatiota on olemassa kolme kappaletta:

- $has_one (Esim. DataObject kuuluu yhteen sivutyyppiin)
- $has_many (Esim. DataObject kuuluu moneen sivutyyppiin)
- $many_many (Esim. DataObject kuuluu moneen DataObjectiin ja sama relaatio myös toisinpäin). (Silverstripe.org 2015e.)

```
class Car extends DataObject{

    private static $db = array(
        "Model" => "Varchar(255)",
        "Color" => "Varchar(255)",
        "Year" => "Int",
        "Manufacturer" => "Varchar(255)",
        "Price" => "Int"
    );

    private static $has_one = array(
        "CarDealerPage" => "CarDealerPage"
    );

}
```

Esimerkkikoodi 6. DataObjectin perusrakenne

Jotta tietokannan tiedot päivittyisivät uuden DataObjectin tiedoilla, täytyy selaimessa lisätä komento */dev/build/* sivun linkin perään ja ajaa se eli esim. USBWebserverin osoitteesta tulee *localhost:8080/dev/build/.* Hyvänä käytäntönä on lisätä myös muuttuja *?flush=1* linkin loppuun, jolloin Silverstripe tyhjentää samalla välimuistinsa. Tällöin USBWebserverin linkki näyttäisi seuraavanlaiselta: *localhost:8080/dev/build/?flush*=1. Tämän jälkeen Silverstripe pyytää sisäänkirjautumista. Sisäänkirjautumisen jälkeen Silverstripe näyttää tietokannan päivittyneen oikein (Kuvio 8) tai ilmoittaa sattuneesta virheestä ja keskeyttää toiminnon. (Silverstripe.org 2015e.)



Kuvio 8. Tietokannan päivittyminen */dev/build/*-komennon ajamisen jälkeen

2.4 Template

Silverstripen template eli aihio on yksinkertaisesti tekstitiedosto, jolla on ss-tiedostopääte. Tiedostot voivat käyttää monia ohjelmointikieliä kuten HTML, XML, CSV, JSON jne. Silverstripen SSViewer-luokka käy kaikki ss-tiedostot läpi ja jäsentää ne esim. HTML-ohjelmointikieleksi. Tällöin voi käyttää Silverstripen omaa syntaksia ja sisäänrakennettuja muuttujia. Niillä voi hoitaa konditionaalista logiikkaa, käydä läpi listoja jne. ilman PHP-ohjelmointia. Esimerkkikoodi 7 näyttää oletusprojektin pää-templaten eli Page.ss-tiedoston rakenteen. (Silverstripe.org 2015c.)

```
<!--[if !IE]><!-->
<html lang="$ContentLocale">
<!--<![endif]-->
<!--[if IE 6 ]><html lang="$ContentLocale" class="ie ie6"><![endif]-->
<!--[if IE 7 ]><html lang="$ContentLocale" class="ie ie7"><![endif]-->
<!--[if IE 8 ]><html lang="$ContentLocale" class="ie ie8"><![endif]-->
<head>
    <% base_tag %>
    <title><% if $MetaTitle %>$MetaTitle<% else %>$Title<% end_if %> &raquo; $SiteConfig.Title</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    $MetaTags(false)
    <!--[if lt IE 9]>
    <script src="//html5shiv.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
    <% require themedCSS('reset') %>
    <% require themedCSS('typography') %>
    <% require themedCSS('form') %>
    <% require themedCSS('Layout') %>
    <link rel="shortcut icon" href="$ThemeDir/images/favicon.ico" />
</head>
<body class="$ClassName<% if not $Menu(2) %> no-sidebar<% end_if %>" <% if $i18nScriptDirection %>dir="$i18nScriptDirection"<% end_if %>>
<% include Header %>
<div class="main" role="main">
    <div class="inner typography line">
        $Layout
    </div>
</div>
<% include Footer %>

<% require javascript('framework/thirdparty/jquery/jquery.js') %>
<%-- Please move: Theme javascript (below) should be moved to mysite/code/page.php  --%>
<script type="text/javascript" src="{$ThemeDir}/javascript/script.js"></script>

</body>
</html>
```

Esimerkkikoodi 7. Page.ss-tiedosto esimerkkinä templatesta

Templaten muuttujat muistuttavat PHP:n muuttujia. Niillä on edessä aina $-merkki eli esim. $Layout tai $Title. Silverstripessa on paljon sisäänrakennettuja muuttujia, jotka ovat globaaleja eli niitä voi käyttää joka sivulla. Yleisesti muuttujat tulevat joko mysite-kansion samannimisen PHP-tiedoston staattisesta $db-taulukkomuuttujasta tai Controllerista, jota käsitellään luvussa 2.5. Muuttujia voi kutsua kolmella tavalla:

- $Title, jolloin Silverstripe hakee tiedon joko $db-taulukkomuutujasta tai kutsuu Controllerissa olevaa Title()-metodia ja palauttaa tuloksen.
- $Title(param), jolloin Silverstripe kutsuu Controllerissa olevaa Title()-metodia syötetyllä parametrilla ja palauttaa tuloksen.
- $Title.Long, jolloin Silverstripe kutsuu Controllerissa olevaa Title()-metodia ja sen jälkeen vielä Title()-metodin palautetussa tuloksessa olevaa Long()-metodia ja palauttaa tuloksen. (Silverstripe.org 2015c.)

Konditionaalista logiikkaa voi templatessa hyödyntää käyttämällä muista ohjelmointikielistä tuttua if-else-rakennetta tai if-else_if-else-rakennetta (Esimerkkikoodi 8).

```
<% if [logiikka tai muuttuja, jota testataan] %>
    <!-- KOODI --->
<% else_if  [logiikka tai muuttuja, jota testataan] %>
    <!-- KOODI --->
<% else %>
    <!-- KOODI --->
<% end_if %>
```

Esimerkkikoodi 8. Silverstripe templaten if-else-rakenne

Listoja, jotka ovat tyyppiä DataList tai ArrayList, voi käydä läpi käyttämällä loop-rakennetta. Loop-rakenteen logiikka on sama kuin muissakin silmukkaratkaisuissa. Rakenne käy läpi kaikki listassa olevat elementit ja tulostaa kyseessä olevan elementin tiedot, jotka on määritelty loop-rakenteen näkyvyysalueen sisällä. Esimerkkikoodi 9 näyttää $Cars-listan, joka saattaa sisältää satojakin elementtejä. Lista voidaan käydä läpi ja tulostaa kaikki tarvittavat tiedot käyttämällä vain muutamaa koodiriviä. (Silverstripe.org 2015c.)

```
<ul>
    <% loop $Cars %>
        <li>
            <div class="car-info-container">
                <h2>$Model</h2>
                <h3>$Year</h3>
                <h3>$Price</h3>
            </div>
        </li>
    <% end_loop %>
</ul>
```

Esimerkkikoodi 9. Loop-rakenne

Templaten ylin näkyvyysalue on sen sivun Page_Controller, joka löytyy samannimisestä PHP-tiedostosta kuin template. Esimerkiksi Page.ss-tiedoston Page_Controller löytyy mysite-kansion Page.php-tiedostosta. Controllerit käsitellään tarkemmin seuraavassa luvussa. Muuttujia ja metodeita pystytään kutsumaan vain siitä näkyvyysalueelta, josta kutsu tehdään. Näkyvyysaluetta pystytään vaihtamaan käyttämällä muuttujia $Up ja $Top. $Up menee aina yhden näkyvyysaluetason ylöspäin, ja sen pystyy linkittämään monta kertaa peräkkäin, jos on tarvetta mennä useampi taso ylöspäin (esim. $Up.Up.CarModel). Käyttämällä $Top-muuttujaa voidaan helposti mennä ylimpään tasoon. Näkyvyysaluetta pystyy myös muuttamaan käyttämällä with-rakennetta, jolloin vältytään toistamasta samaa koodia valitsemalla tietty olio näkyvyysalueeksi esimerkkikoodi 10 mukaisesti. (Silverstripe.org 2015c.)

```
<!-- ILMAN WITH-RAKENNETTA -->
<div>
        <p>$Car.Model</p>
        <p>$Car.Year</p>
        <p>$Car.Price</p>
</div>

<!-- WITH-RAKENTEEN KANSSA -->
<div>
    <% with $Car %>
        <p>$Model</p>
        <p>$Year</p>
        <p>$Price</p>
    <% end_with %>
</div>
```

Esimerkkikoodi 10. With-rakenne

2.5 Controller

Controller-luokan tehtävänä on käsitellä HTTP:n vastaukset ja pyynnöt, jotka ovat Silverstripessa SS_HTTPRequest ja SS_HTTPResponse. Controller käsittelee erilaiset lomakkeet, tulostaa oikeat templatet ja tarkistaa sekä ajaa käyttäjäoikeuksiin perustuvat toiminnot. Yleisin Controller on Page_Controller, joka kuuluu jokaiseen sivutyyppiin. (Silverstripe.org 2015f.)

Page_Controller hakee templaten, jolla on sama nimi kuin sillä ja sitoo sen oikeaan sivutyyppiin. Tämä reititys tapahtuu automaattisesti, koska sisällönhallintajärjestelmä hoitaa sen aina, kun sivut on luotu tai sivutyyppejä on käytetty Admin-paneelin kautta. (Silverstripe.org 2015g.)

Kaikki Controllerissa määritellyt metodit ovat ns. actioneita eli toimintoja, joita pystyy ajamaan suoraan selaimessa kirjoittamalla toiminnon nimi oikean sivutyypin linkin loppuun. Jotta peruskäyttäjällä ei olisi samoja oikeuksia kuin järjestelmävalvojalla, Silverstripe rajoittaa toimintojen käytön ellei niitä erikseen sallita. Tämä salliminen tapahtuu Controllerin staattisessa $allowed_actions-muuttujassa, jossa määritellään taulukko kaikista sallituista toiminnoista ja tarvittaessa määritellään, kuka niitä pystyy ajamaan selaimessa. Esimerkkikoodi 11 esittää, miten muuttuja $allowed_actions voidaan määritellä. (Silverstripe.org 2015h.)

```
private static $allowed_actions = array(
    'submit',
    'delete',
    'AddPage',
    'DeletePictures',
    'UpdateCart',
    'deleteAllComments' => 'ADMIN',
    'replacePicture'
);
```

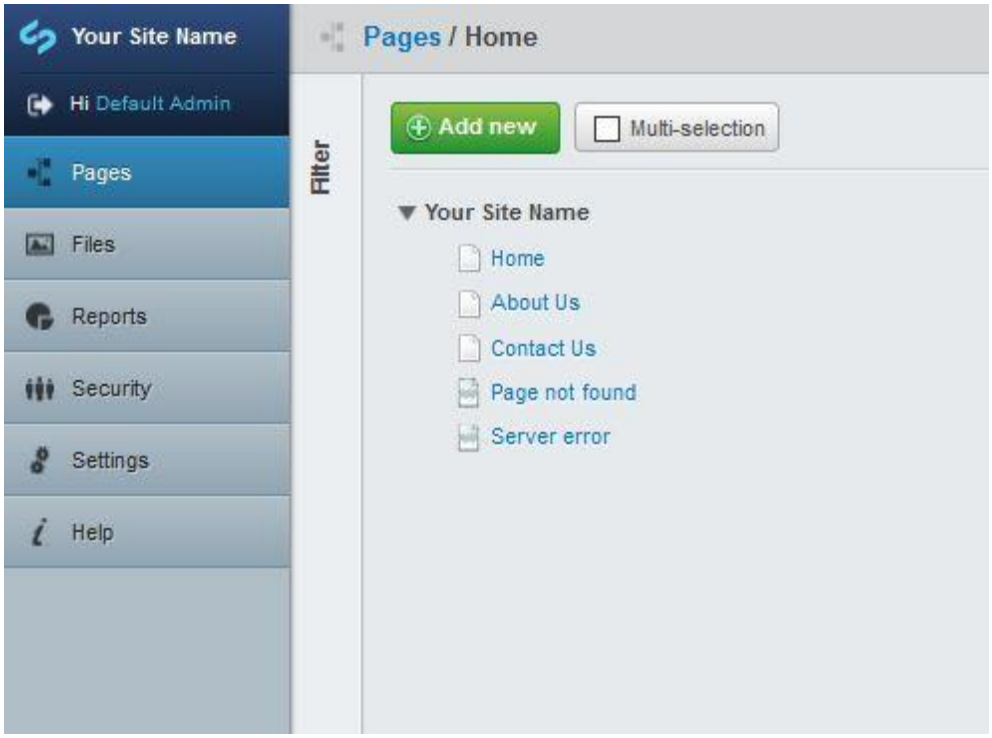
Esimerkkikoodi 11. Controllerin staattinen $allowed_actions-muuttuja

2.6 Admin-paneeli

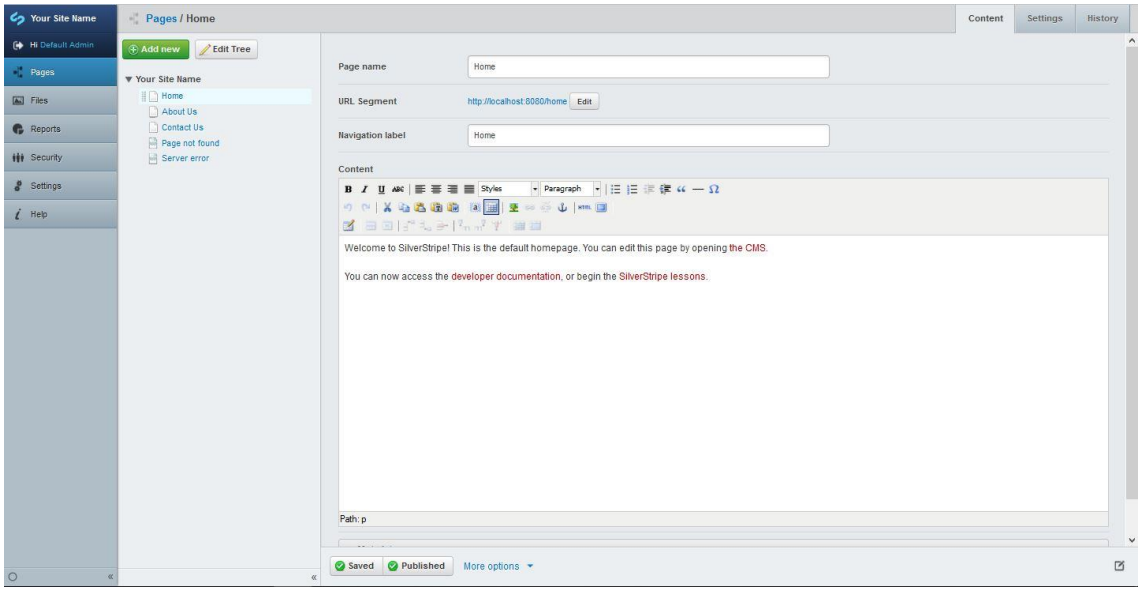
2.6.1 Rakenne

Admin-paneelissa käyttäjä lisää sivustolle sisältöä ja muokkaa sitä. Paneeliin pääsee ajamalla selaimessa osoite *localhost:8080/admin*, jonka jälkeen pitää kirjautua järjestelmävalvojan tunnuksilla sisälle. Järjestelmävalvojan tunnukset luotiin asennuksen yhteydessä. Kun kirjautuminen on onnistunut, käyttäjä näkee ensimmäisenä sivuston sivurakenteen eli ns. sivupuun (Kuvio 9). Painamalla mitä tahansa sivua esiin tulee Silverstripe-sisällönhallintajärjestelmän muokkaustyökalut (Kuvio 10). Admin-paneelissa pystyy sivujen lisäyksen ja muokkauksen lisäksi määrittelemään sivuston asetukset, lisäämään erilaiset tiedostot myöhempää käyttöä varten, määrittelemään turvallisuusasetukset sekä lukemaan erilaiset sivustoa koskevat raportit. Jokaisella sivulla on omat oletusvälilehdet, joita ovat Content (sisältö), Settings (asetukset) ja History (sivun historia). (Silverstripe.org 2015i.)

Admin-paneeli on jaettu kahteen osaan. Vasemmalla olevaa alue, johon kuuluvat menu ja sivupuu, on nimeltään Left ja kaikki muu Main. Tällöin Controller-luokka, joka hallitsee Admin-paneelin sisältöä, on nimeltään LeftAndMain. Kaikki LeftAndMain-aliluokat näkyvät automaattisesti päävalikossa. ModelAdmin-luokka on yleisin LeftAndMain-aliluokka, jota käyttäjä tulee mahdollisesti käyttämään. Luodakseen uuden menu-osion käyttämällä ModelAdminia pitää tehdä DataObject, jota ModelAdmin tulee käyttämään. Sen jälkeen voidaan tämä DataObject hyödyntää ModelAdminissa kuten esimerkkikoodi 12 näyttää. Ajamalla selaimessa osoite *localhost:8080/dev/build/?flush=1* saadaan uusi menu-osio näkymään Admin-paneelissa. (Silverstripe.org 2015j.)



Kuvio 9. Admin-paneelin sivupuu



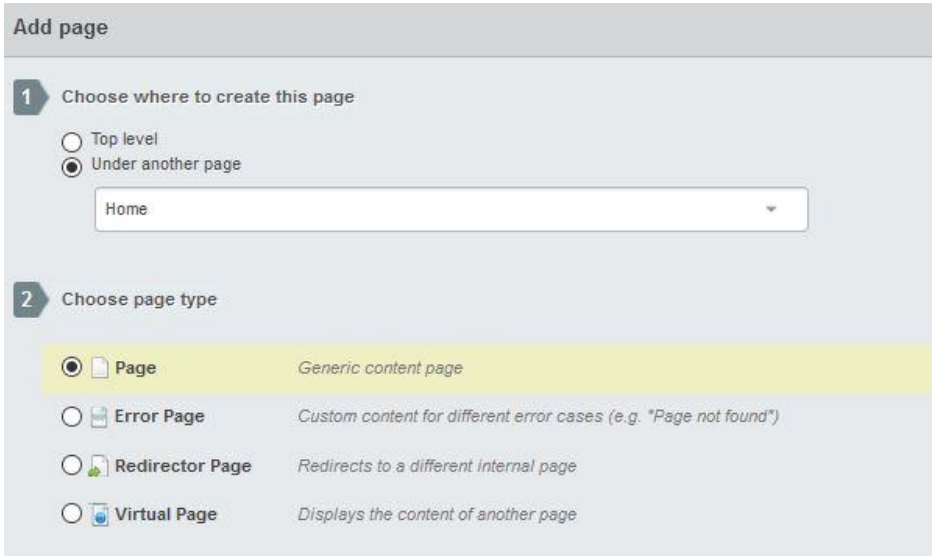
Kuvio 10. Admin-paneelin muokkaustyökalut

```
class CarAdmin extends ModelAdmin {

    private static $menu_title = 'Cars';

    private static $url_segment = 'cars';

    private static $managed_models = array (
        'Car'
    );
}
```

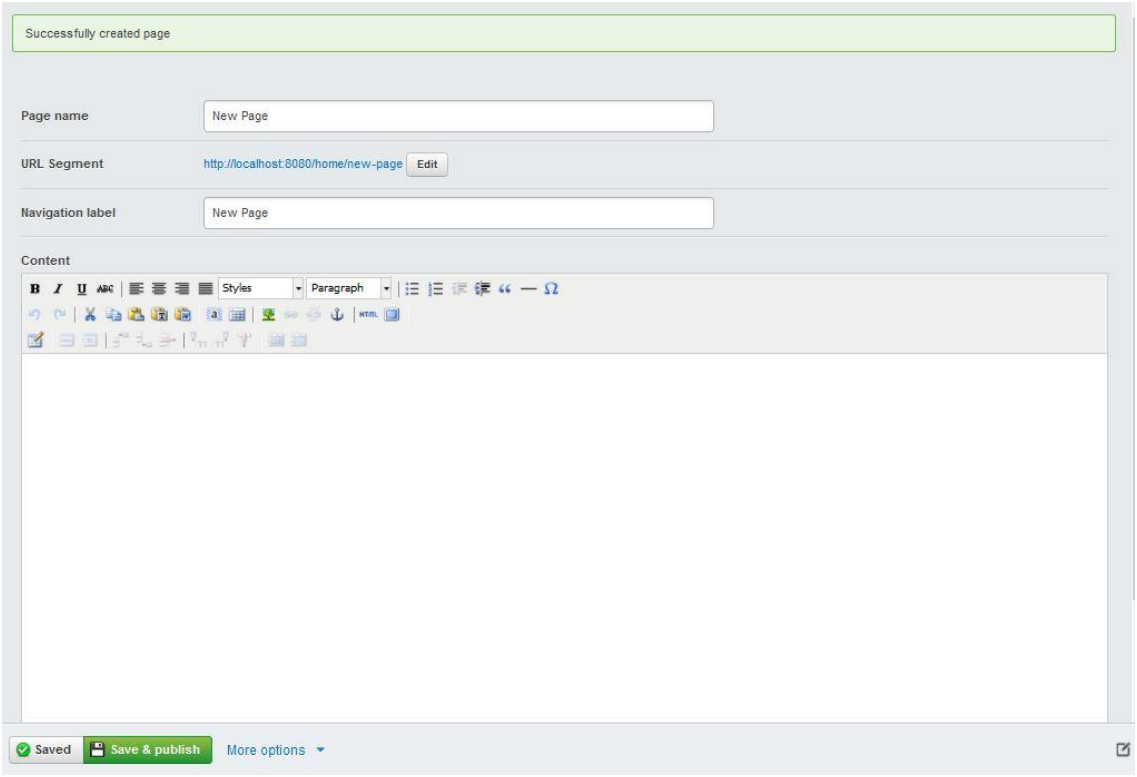
Esimerkkikoodi 12. ModelAdmin-esimerkki

2.6.2 Sivujen lisääminen

Sivuja voidaan lisätä painamalla Add New-painiketta sivupuunäkymässä tai klikkaamalla oikealla hiiren painikkeella mitä tahansa sivupuun sivua ja valitsemalla valikosta *Add new page here* se sivutyyppi, joka halutaan lisätä. Jos sivu lisätään Add New-painikkeesta, käyttäjä siirtyy Add page-sivulle, jossa aluksi valitaan paikka, jonne sivu luodaan ja sitten valitaan sivutyyppi uudelle sivulle (Kuvio 11). Kun molemmat vaiheet on suoritettu, sivu voidaan luoda painamalla Create-painiketta. Sen jälkeen käyttäjä siirtyy uuden sivun muokkaustyökaluun, jossa hän voi määritellä sivun osoitteen, otsikon, navigaationimikkeen sekä lisätä pääsisältö (Kuvio 12). Kun kaikki tiedot on syötetty, sivu voidaan joko julkaista kaikkien nähtäväksi painamalla Save & Publish -painiketta tai pelkästään tallentaa luonnos myöhempää muokkausta varten. (Silverstripe.org 2015i.)



Kuvio 11. Uuden sivun lisääminen Admin-paneelissa



Kuvio 12. Sivun muokkaustyökalu Admin-paneelissa

3 OMIEN DATAOBJECTIEN LUOMINEN JA KÄYTTÄMINEN

3.1 DataObjectin ohjelmointi

DataObjectin luominen alkaa lisäämällä PHP-luokka mysite-kansioon. Luokalle täytyy lisätä laajennus DataObject. Tässä luvussa luodaan DataObject nimeltään Phone. Kun perusluokka on luotu, määritellään kaikki luokan muuttujat staattisessa $db-muuttujassa, jotta tiedot tallentuisivat tietokantaan. Tämä luokka sisältää seuraavat muuttujat:

- Model, tietotyyppiä Text
- Manufacturer, tietotyyppiä Text
- Price, tietotyyppiä Double
- ManufacturingDate, tietotyyppiä Date
- Description, tietotyyppiä Text.

Esimerkkikoodi 13 näyttää, miten tämä on toteutettu.

```php
<?php

class Phone extends DataObject{

    private static $db = array(
        "Model" => "Text",
        "Manufacturer" => "Text",
        "Price" => "Double",
        "ManufacturingDate" => "Date",
        "Description" => "Text"
    );

}
```

Esimerkkikoodi 13. Phone-DataObjectin muuttujat

Sen jälkeen DataObject pitää sitoa johonkin tiettyyn sivuun tai toiseen DataObjectiin, jotta tietokannan relaatiot toimisivat oikein. Tämä DataObject saa esiintyä joka sivulla, joten se sidotaan Page-sivutyyppiin. Page-luokka on kaikkien sivutyyppien yliluokka ja sen takia tällaisen relaation omaava DataObject saadaan halutessa tulostettua joka sivuston sivulla. Koska halutaan, että Phone-DataObjectin relaatio olisi vain Page-luokalla, sitominen tehdään staattisessa $has_one-

muuttujassa. Koska Page-luokalla tulee olemaan monta Phone-DataObjectia, myös Page-luokkaan täytyy luoda relaatio, joka tehdään staattisessa $has_many-muuttujassa (Esimerkkikoodi 14). Esimerkkikoodissa vasen Phones-arvo on taulukko, joka tulee sisältämään kaikki luodut Phone-DataObjectit, jotka määritellään oikealla olevalla Phone-arvolla.

```php
<?php
class Page extends SiteTree {

    private static $has_many = array(
        "Phones" => "Phone"
    );

}
```

Esimerkkikoodi 14. Page-luokan has_many-relaatio

Kun relaatiot on luotu, seuraavaksi pitää luoda kentät Admin-paneelia varten, jotta pystytään lisäämään sisällöt DataObjectiin. Tämä tapahtuu lisäämällä metodi getCMSFields() DataObjectiin. Tässä metodissa määritellään kaikki kentät, jotka tulevat käyttämään $db-taulukkomuuttujassa määritetyt muuttujat. Admin-paneelin kentät määritellään getCMSFields()-metodissa olevassa FieldList-tyyppisessa muuttujassa, joka on yleensä nimeltään $fields. Kentät lisätään tähän muuttujaan siinä järjestyksessä, jossa halutaan niiden esiintyvän Admin-paneelissa. Phone-DataObjectissa käytetään seuraavia kenttiä:

- TextField
- DateField
- TextareaField.

Kentissä pystyy määrittelemään myös niitä kuvaava teksti sekä tilapäinen teksti, joka tulee näkymään kentässä DataObjectin luomisen yhteydessä. Tämä kaikki syötetään kenttien create()-metodin parametreihin. Ensimmäiseen parametriin kirjoitetaan $db-taulukkomuuttujassa määritelty muuttuja, joka halutaan käyttää. Toiseen parametriin kirjoitetaan kentän kuvaus ja kolmanteen tilapäinen teksti, joka halutaan näkymään kentässä oletuksena. Esimerkkikoodi 15 näyttää, miten tämä on toteutettu.

```
public function getCMSFields() {
    $fields = FieldList::create(
        TextField::create('Model', 'Puhelimen malli'),
        TextField::create('Manufacturer', 'Valmistaja', 'Nokia'),
        TextField::create('Price', 'Hinta'),
        DateField::create('ManufacturingDate', 'Valmistuspäivämäärä'),
        TextareaField::create('Description', 'Puhelimen kuvaus')
    );

    return $fields;
}
```

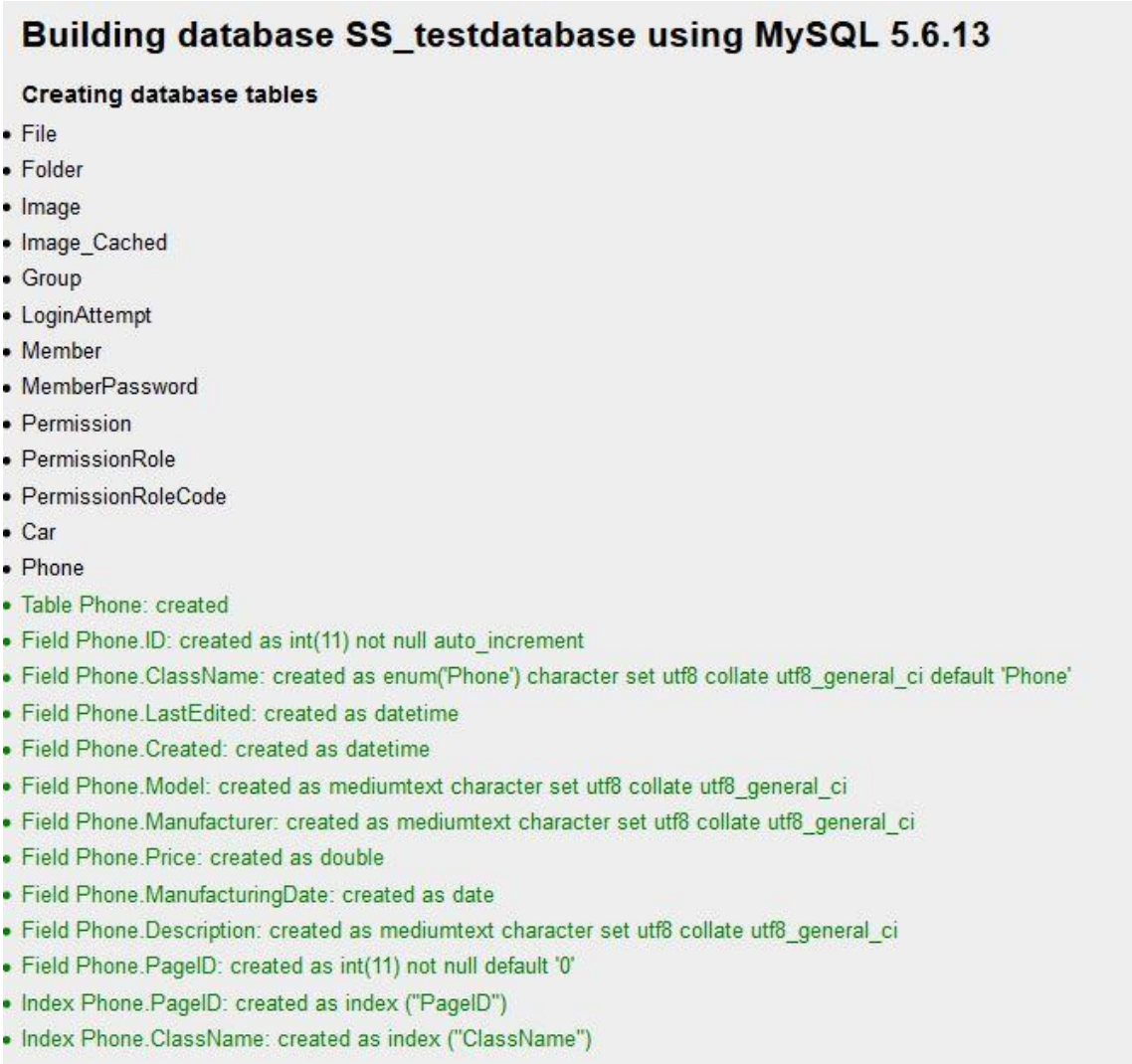
Esimerkkikoodi 15. Phone-DataObjectin getCMSFields()-metodi

Page-luokkaan pitää myös lisätä getCMSFields()-metodi, jotta kaikki sen sivun DataObjectit olisivat hallittavissa yhdestä paikasta. Koska Page-luokka sisältää useamman DataObjectin, käytetään GridField-kenttää, joka hyödyntää aikaisemmin luotua Phones-taulukkoa Page-luokan $has_many-muuttujassa. GridField listaa kaikki taulukon DataObjectit ja se sisältää lisäys-, muokkaus- ja poistotoiminnallisuudet. Jos halutaan erottaa GridField-kenttä muista kentistä, se voidaan siirtää omaan välilehteen käyttämällä FieldListin addFieldToTab()-metodia, johon määritellään välilehden nimi sekä kenttä, jonka se tulee sisältämään. Koska kyseessä on sivu, FieldListia ei tarvitse luoda erikseen. Se voidaan hakea yliluokasta ja tallentaa $fields-muuttujaan komennolla parent::getCMSFields(). Esimerkkikoodi 15 esittää Page-luokan getCMSFields()-metodin.

```
public function getCMSFields() {
    $fields = parent::getCMSFields();
    $fields->addFieldToTab('Root.Phones', GridField::create(
        'Phones',
        'Phones on this page',
        $this->Phones(),
        GridFieldConfig_RecordEditor::create()
    ));

    return $fields;
}
```

Esimerkkikoodi 16. Page-luokan getCMSFields()-metodi

Kun kaikki tämä on tehty, DataObject voidaan lisätä tietokantaan ajamalla selaimessa osoite *localhost:8080/dev/build/?flush=1*, jonka jälkeen taulukot ja niiden relaatiot luodaan tietokantaan (Kuvio 13).



Kuvio 13. Phone-DataObjectin luominen

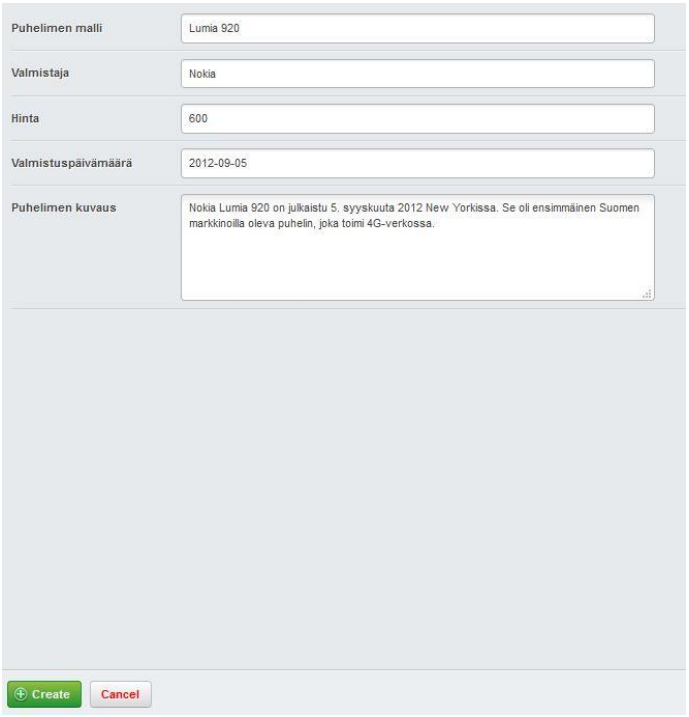
3.2 DataObjectin käyttäminen Admin-paneelissa

Kun DataObject on luotu tietokantaan, seuraava vaihe on luoda oliot siitä Admin-paneelissa. Jokaisella Page-tyyppisellä sivulla on nyt Phones-niminen välilehti, jossa voidaan luoda sen sivun Phone-DataObjectit (Kuvio 14). Välilehteä painamalla käyttäjä siirtyy viime luvussa luotuun GridField-kenttään, jossa DataObjectit luodaan. Painamalla Add phone-painiketta voidaan aloittaa luomisprosessi (Kuvio 15).

Kuvio 14. Phones-välilehti Admin-paneelissa

Kuvio 15. GridField-kenttä Phone-DataObjecteja varten

Add phone-painiketta painalluksen jälkeen esiin tulee lomake sisältäen kentät, jotka luotiin Phone-DataObject-luokassa. Kun kaikki tiedot on syötetty, DataObject voidaan luoda painamalla Create-painiketta (Kuvio 16). Samaa prosessia voidaan toistaa niin monta kertaa kuin on tarpeen. Tässä työssä on luotu neljä DataObjectia: Nokia Lumia 920, Iphone 6S, Samsung Galaxy S4 ja Sony Xperia Z (Kuvio 17).



Kuvio 16. Phone-DataObjectin luominen Admin-paneelissa

Kuvio 17. GridField-kenttä neljällä Phone-DataObjectilla

Jos halutaan lisää informaatiota GridFieldiin selkeyttämään luettavuutta, voidaan lisätä Phone-DataObject-luokan staattiseen $summary_fields-muuttujaan ne kentät, joista halutaan yhteenvetosarake GridFieldiin (Esimerkkikoodi 17). Kun selaimessa ajetaan osoite *localhost:8080/dev/build/?flush=1*, päivittyy myös GridField uusilla yhteenvetosarakkeilla (Kuvio 18). DataObjecteja voi muokata painamalla sivu-ikonia GridField-rivin oikeassa kulmassa tai painamalla kyseessä olevaa GridField-riviä. Poistaminen tapahtuu joko painamalla GridField-rivin oikeassa kulmassa olevaa punaista rasti-ikonia tai menemällä kyseessä olevan GridField-rivin muokkaustilaan ja painamalla Delete-painiketta.

```
public static $summary_fields = array(
    'Manufacturer' => 'Manufacturer',
    'Model' => 'Model',
    'Price' => 'Price'
);
```

Esimerkkikoodi 17. Phone-DataObjectin $summary_fields-muuttuja

Kuvio 18. GridField-kenttä uusilla yhteenvetosarakkeilla

3.3 DataObjectin käyttäminen templatessa

Kun kaikki DataObjectit ovat luotu Admin-paneelissa, seuraava vaihe on tulostaa ne kaikkien nähtäväksi templatessa. Koska Phone-DataObjectit ovat tallennettu Phones-taulukkoon, sen läpikäymiseen pitää käyttää loop-rakennetta. Esimerkkikoodi 18 näyttää, miten tämä on toteutettu Page.ss-template-tiedostossa.

```
<% include Header %>
<div class="main" role="main">
    <div class="inner typography line">
        $Layout

        <% loop Phones %>
            <div>
                <h2>Puhelin: $Manufacturer $Model</h2>
                <h3>Hinta: $Price€</h3>
                <h3>Julkaisupäivämäärä: $ManufacturingDate.Nice</h3>
                <p>$Description</p>
            </div>
        <% end_loop %>

    </div>
</div>
<% include Footer %>
```

Esimerkkikoodi 18. Phones-taulukon DataObjectien tulostaminen templatessa

Loop-riviin sisälle kirjoitetaan taulukon nimi ja rakenteen sisälle kirjoitetaan vain yhden sivuelementin runko, joka sisältää ne DataObjectin tiedot, jotka halutaan tulostaa. Tiedot haetaan DataObjectin muuttujilla, jotka templatessa alkavat $-merkillä. Koska päivämäärässä on käytetty Date-tyyppistä muuttujaa, voi käyttää Date-luokan Nice()-metodia päivämäärän esittelytavan muuttamiseen.

Loop-rakenne käy kaikki Phones-taulukossa olevat DataObjectit läpi ja tulostaa ne peräkkäin. Jotta uudet elementit näkyisivät sivulla, pitää ajaa selaimessa sen sivun osoite, johon halutaan tulostaa elementit ja lisätä osoitteen perään komento *?flush=all*, joka tyhjentää kyseessä olevan sivun Silverstripe-välimuistin. Esimerkiksi kotisivun välimuisti tyhjennetään osoitteella *localhost:8080/home/?flush=all*. Tämän jälkeen elementit näkyvät sivulla (Kuvio 19).

Kuvio 19. Phone-DataObjectien tulostaminen sivulle järjestyksessä

4 POHDINTA

Silverstripe on vielä suhteellisen tuntematon alusta sisällönhallintajärjestelmien markkinoilla, vaikka se on ollut olemassa jo useita vuosia. Tämän työn tavoitteena oli ohjata aloittelevaa kehittäjää Silverstripen asennuksessa, käyttöönotossa sekä perusohjelmoinnissa luomalla ohje Silverstripen keskeisimmistä osista. Kyseinen tavoite saavutettiin.

Silverstripe on osoittautunut todella helppokäyttöiseksi kokonaisuudeksi verrattuna sisällönhallintajärjestelmiin WordPress tai Joomla, koska esimerkiksi HTML-koodiin ei tarvinnut sekoittaa PHP-koodia. Silverstripe-syntaksin käyttäminen sivuaihioissa helpotti koodin lukemista, kirjoittamista sekä sen jäsentelyä.

Opinnäytetyön tekeminen oli haastava prosessi, koska olen tehnyt sen oman yrityksen projektien ohella ja joskus opinnäytetyön sovittaminen aikatauluihin oli haasteellista. Aihe oli kuitenkin ennestään tuttu, mikä helpotti kirjoittamisprosessia.

Tässä opinnäytetyössä ei käsitelty kaikkia Silverstripen osia vaan perehdyttiin perusprojektin olennaisiin osiin. Jatkossa opasta voisi kehittää lisäämällä tietoa Extension-tyyppisistä laajennuksista, lomakkeista, sivuston testaustyökaluista, kieliversioinnista sekä sivuston turvallisuuden varmistamisesta.

LÄHTEET

Silverstripe.org 2015a. Download Page. Viitattu 18.10.2015 http://www.silverstripe.org/download/

- 2015b. Directory Structure. Viitattu 19.10.2015 https://docs.silverstripe.org/en/3.2/getting_started/directory_structure/

- 2015c. Template Syntax. Viitattu 19.10.2015 https://docs.silverstripe.org/en/3.2/developer_guides/templates/syntax/

- 2015d. Working with Multiple Templates. Viitattu 19.10.2015 http://www.silverstripe.org/open-source/learn/lessons/working-with-multiple-templates

- 2015e. Introduction to the Data Model and ORM. Viitattu 20.10.2015 http://www.silverstripe.org/open-source/learn/lessons/working-with-multiple-templates

- 2015f. Controllers. Viitattu 26.10.2015 https://docs.silverstripe.org/en/3.2/developer_guides/controllers/

- 2015g. Introduction to Controllers. Viitattu 26.10.2015 https://docs.silverstripe.org/en/3.2/developer_guides/controllers/introduction

- 2015h. Access Control. Viitattu 26.10.2015 https://docs.silverstripe.org/en/3.2/developer_guides/controllers/access_control/

- 2015i. Using the CMS. Viitattu 27.10.2015 https://docs.silverstripe.org/en/3.2/tutorials/building_a_basic_site/

- 2015j. CMS architecture. Viitattu 27.10.2015 http://www.silverstripe.org/learn/lessons/introduction-to-modeladmin